International Journal of Computer Networking,
Wireless and Mobile Communications (IJCNWMC)
ISSN (P): 2250-1568; ISSN (E): 2278-9448
Vol. 9, Issue 1, Jun 2019, 35-44

OPTIMIZING IoT CLOUD ARCHITECTURES FOR PIPELINING DATA THROUGH MACHINE LEARNING MODELS

DIPANJAN DE & JABANJALIN HILDA. J

School of Computer Science and Engineering, Vellore Institute of Technology, Vellore, Tamil Nadu, India

ABSTRACT

Rapid advancements in Internet of Things enable meaningful solutions to several problems, which were never even thought of, until recently. Machine Learning, Deep Learning and Neural Networks play a vital role in this trend, and most of the IoT platforms, pipeline data through models, for a number of reasons, including but not limited to threat detection, real time analytics, disaster prediction, etc. These models require an enormous amount of computing resources, which typically requires a GPU, and integrating these models with the cloud poses a number of major challenges involving computing paradigm of the cloud. In this paper, we propose several optimized solutions to these problems faced while pipelining enormous amount data through models in real-time. These solutions addresses issues of scalability as well, when the platform provider needs to expand and will require increased number of pipelines in real time.

KEYWORDS: IoT, Cloud, Machine Learning, Deep Learning, Neural Networks, GPU & API

Received: Mar 13, 2019; **Accepted:** Apr 03, 2019; **Published:** Apr 27, 2019; **Paper Id.:** IJCNWMCJUN20195

Original Article

1. INTRODUCTION

Internet of Things is rapidly evolving and advancing and a majority of its operations are dependent on Machine Learning, Deep Learning and Neural Networks. They enable IoT platforms to implement a number of important features, which are delivered to the customers. Several important real time tasks such as Abnormal Behaviour Profiling of IoT Devices [1], Threat Detection [2] and others that are vital to cloud platforms powering its devices, all depend on them. Furthermore, protecting models from poisoning attacks also can depend on dedicated machine learning models that detect and mitigate these attacks [3].

Pipelining API data through these models in most cases, require a GPU, otherwise it would take an awful lot of time to process the data, resulting in delayed API responses. Most IoT platforms rely on a dedicated or third party cloud, which provides dedicated machine learning VM instances powered by a GPU to pipeline the API data. These GPUs enable matrix operations to be carried out in parallel, and these can be performed in distributed systems to increase efficiency [4]. The main objective of these workflows is to reduce API response times, which is vital for any platform.

In this paper, we present various methods of optimizing models, pipeline data, re-train and update them. In section 2, we go through various promising cloud architectures for handling multiple API requests at the same time, which involves pipelining through models, security checks and database operations. We go through various algorithms to update and re-train machine-learning models rapidly, and talk about data clustering in section 3. We display our results and observations in section 4 and conclude our paper and talk about future scope and enhancements in section 5.

2. CLOUD ARCHITECTURE

It is estimated that more than 20 billion IoT devices will be connected to the internet by the year 2020. Setting up the right cloud architecture to handle such enormous amount of data and API requests originating from these devices, is an essential ingredient to improve performance in the cloud.

One way to further enhance the availability of the API endpoints is by incorporating load balancers [5] [6]. It distributes the traffic across a cluster of servers and reduces load on individual servers and prevents any one server from becoming a single point of failure, which may cripple the entire platform. To implement this, we would of course require a server cluster, with several individual servers located at multiple locations. Such load balancers can also be implemented while pipelining data through machine learning models in individual servers in the cluster, which may run pipelines in parallel in the same server or may call separate instances in a different server cluster. This workflow will significantly reduce the response time.

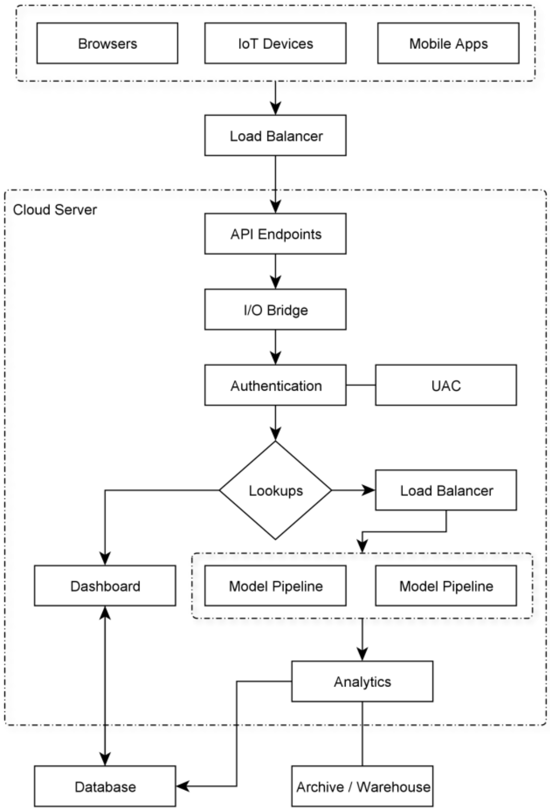

Figure 1: IoT Cloud Platform Architecture

One such architecture is depicted in Figure 1, where the API requests are handled through a load balancer and distributed to a server cluster. The request passes through I/O bridge, which is a Publish-Subscribe Broker, which then authenticates and forwards the request to the intended destination via lookup tables. We will describe each of these components in detail.

2.1. I/O Bridge

The I/O Bridge is a Publish-Subscribe broker, which increases the overall security, efficiency and scalability of the system [7]. The bridge accepts HTTP requests and acts as a message broker when any two systems or sub-systems

wants to communicate with each other. It acts as an asynchronous service-to-service communication.

I/O Bridge finds the Subscriber of the message through the help of lookup tables, which can be predefined or dynamically generated and updated, in cases when IP or port of the sub-systems are subject to changes, for instance, when floating IP is incorporated.

2.2. Payload Handling

A cloud platform may encounter and have to deal with hundreds, if not thousands of sensor data transmissions at the same time. To deal with such enormous amount of request payloads, we can use a slightly different approach depending on use cases, as depicted in Figure 2.

We can primarily divide use cases based on the following aspects:

- Each payload is needed to be pipelined for classification before archiving and:
  - The model needs to be updated via online learning immediately
  - The model can be updated later at a different time via offline or batch/mini-batch learning
- The payload can be archived and later classified when the server has empty task queues at any given time, and then:
  - The model is to be updated immediately via online learning
  - The model can be updated later via offline or batch/mini-batch learning

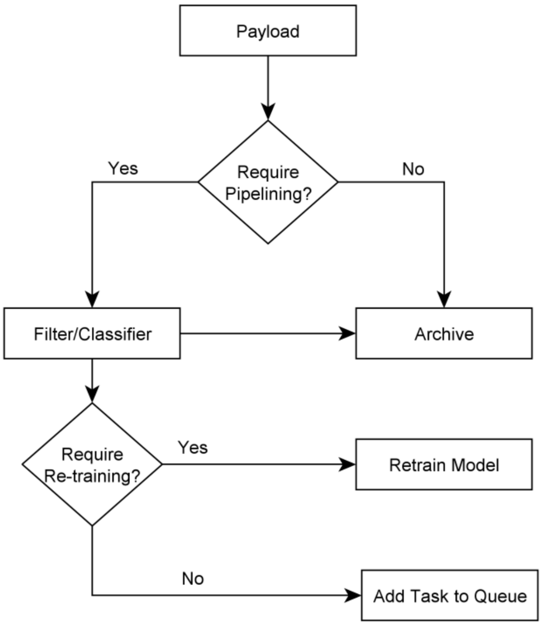

Figure 2: Payload Handling

This approach can save the computational resources for critical tasks at hand, which has to be processed immediately, and when the queue is free, these tasks can be added to the processing queue and completed.

2.3. Moving ML Pipeline to the Front End

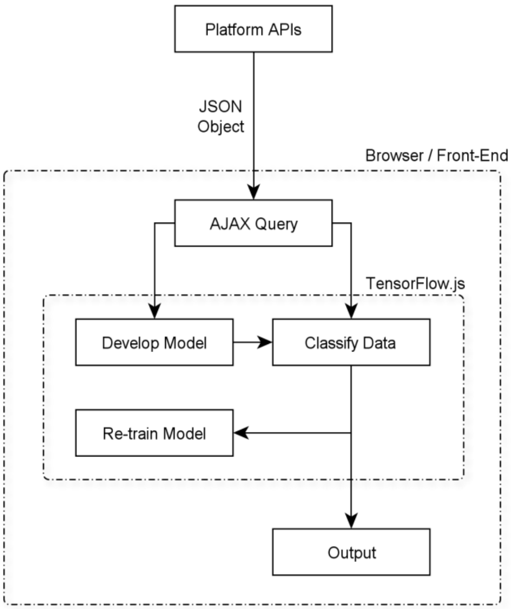

Figure 3: Payload Handling

There are numerous front-end JavaScript libraries and frameworks, which allow pipelining vast amount of data in the client's browsers. In cases where models are okay to be trained after users' discretion, can be done entirely on users' computing resources using JavaScript. Then can even be used to pipeline data, in the front-end entirely.

These conditions are met in many cases such as, for instance, when platforms allow users to write their own models and train and use them on their own dataset. Figure 3 depicts how this works in JavaScript.

This approach can provide additional features to the users, such as ability to develop own models and train them on their own dataset, without leveraging additional resources in the cloud, resulting in a more meaningful and efficient IoT platform.

3. OPTIMIZATION AND RE-TRAINING

In this section, we discuss various optimization techniques, which can help increase efficiency of IoT cloud platforms. We would go through methods of updating or re-training models and how each method can be useful for distinct scenario.

3.1. Online Learning

Existing models can be trained each time a new observation is available, by back propagation. This way, the model adapts locally to the data and is more influenced by recent observations, than the older observations. A great way of re-training models is using Stochastic Gradient Descent [8] [10].

Let us consider, our cost function is

$$min_{\theta}\, J(x, y, \theta)$$

Where, θ is the parameter vector and data streaming in the form of (x^i, y^i). We can update θ using the equation

$$\theta^t = \theta^{t-1} - \nabla_{\theta}\, J(x^i, y^i)$$

This is, in other words, a batch Stochastic Gradient Descent with batch size 1. This method is useful in scenarios where the model needs to adapt to new patters in observations dynamically.

3.2. Data Clustering

Clustering of data before training models is important, and it reduces a significant amount of processing time and increases performance and accuracy of the models [11] [13]. We will take a small example of sensor data measuring temperatures in various locations at different time intervals and see how we can cluster such a diverse dataset.

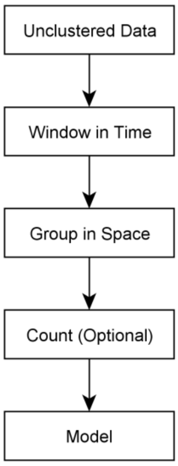

Figure 4: Steps to Cluster Data

Let us consider the following dataset in Table 1. We will cluster the data through the steps depicted in Figure 4 and then forward it to the model.

Table 1: Sample Dataset for Clustering

Latitude	Longitude	Time
X1	Y1	01:01
X1	Y1	01:33
X2	Y2	01:17
X2	Y2	02:15

We will first cluster the dataset based on a different time slots, which gives us the results in Table 2.

```
{[01:00,02:00] -> (X1,Y1)@01:01, (X2,Y2)@01:17, (X1,Y1)@01:33}
```

```
{[02:00,03:00] -> (X2,Y2)@02:15}
```

Table 2: Dataset Clustered by Time Frames

Time Frame	Latitude	Longitude	Time
01:00 – 02:00	X1	Y1	01:01
	X2	Y2	01:17
	X1	Y1	01:33
02:00 – 03:00	X2	Y2	02:15

Then we will group the clustered set in space, based on coordinates, which will give us the results in Table 3.

```
{(X1,Y1),[01:00,02:00] -> (X1,Y1)@01:01, (X1,Y1)@01:33}
```

```
{(X2,Y2),[01:00,02:00] -> (X2,Y2)@01:17}
```

```
{(X2,Y2),[02:00,03:00] -> (X2,Y2)@02:15}
```

Table 3: Dataset Grouped in Space

Coordinates	Time Frame	Data
(X1, Y1)	01:00 – 02:00	(X1,Y1)@01:01
		(X1,Y1)@01:33
(X2, Y2)	01:00 – 02:00	(X2,Y2)@01:17
(X2, Y2)	02:00 – 03:00	(X2,Y2)@02:15

Now our clusters are ready to be provided to the models. Counting the number in the clusters can also help improve performance, if the count is relevant to the model. We would come up with Table 4, if we process the count on our dataset.

```
{(X1,Y1),[01:00,02:00] -> 2

{(X2,Y2),[01:00,02:00] -> 1

{(X2,Y2),[02:00,03:00] -> 1
```

Table 4: Dataset Cluster Count

Coordinates	Time Frame	Count
(X1, Y1)	01:00 – 02:00	2
(X2, Y2)	01:00 – 02:00	1
(X2, Y2)	02:00 – 03:00	1

This clustered data can now be pipelined and the time to process this clustered data will be much lower than the time required to clean, group and then process the original data.

3.3. Offline Learning

This method is applicable when the model needs to be re-trained entirely based on a new and bigger dataset where the new observations have been appended. This can lead to a better global approximation of the target function. This is used in scenarios where there is a fixed dataset or where observations are not added frequently. This method however is not practical for huge datasets.

This approach increases the overall performance of the cloud platform and refrains from pushing the system, when the load on the server is heavy and the API endpoints are handling handful of requests at the same time.

3.4. Batch or Mini-Batch Learning

With this approach, we usually wait until we have a batch of ‘n’ number of observations, and then train the existing model with this batch [14]. The update equation for Batch Learning will be

$$\theta^t = \theta^{t-1} - \Sigma_i \nabla_\theta J(x^i, y^i)$$

Mini Batch is almost the same, except the batch size is smaller, but greater than 1, which is Online Learning.

Incorporating this learning method for updating and re-training the models increase the stability of the model and in the long run, increases accuracy [15]. Processing in batch also results in better performance, if the batches are processed in parallel.

4. PERFORMANCE ANALYSIS

We measured the response time of our architecture involving load balancers and compared it to one, which does not incorporate load balancers. We also considered pipelining data through machine learning models in this test and

compared it to cases where the payload is directly archived, and later pipelined. Results are shown in Table 5.

Table 5: Average Response Time for Processing Payloads

Response Time	Without LB	With LB
Pipelining*	6708ms	5269ms
Direct Archive	376ms	279ms

* Data pipelined through Logistic Regression model without updating or re-training the model


```
POST /gateway/payload HTTP/1.1

HOST: bilonz.com

Content-Type: application/json

Accept: */*

Content-Length: 178

X-Access-Node: 9fc55d56-8cb1-4307-90a5-6ab94d2cc418


{
   "sensors": [
      {
         "id": "8ac48075-f161-4135-a7cc",
         "value": "19.45"
      },
      {
         "id": "0b904768-ef56-4c2a-4gs6",
         "value": "15.12"
      }
   ]
}
```


Figure 5: HTTP Post Payload

For fully utilizing the load balancers, we used three identical servers in a server cluster, and one load balancer for distributing traffic to them uniformly. We sent out five identical payloads, shown in Figure 5, at the same time and measured the response time.

Our observation confirms that in scenarios involving huge traffic, load balancers and choosing whether to pipeline data while handling requests, can help reduce response time significantly.

5. CONCLUSIONS

In this paper, we proposed various techniques of optimizing data pipelining for model analysis and classification. We discussed how choosing the right cloud server architecture can increase platform efficiency and how the right use of load balancers can help distribute traffic across server clusters and improve the response time of the API requests.

We evaluated our architecture performance on various cloud platforms and different types of instances, and we evaluated more than 150 requests involving HTTP GET and POST methods for transferring payload from sensors. Our performance comparison reveals that this architecture can result in almost 21-22% reduction in response time when using load balancers for pipelining and can reduce close to 95% of the response time when direct archived and later processed, instead of immediately pipelining and processing the data through a model. Therefore, configuring the cloud architecture for certain use cases can result in a significant reduction in response time of API requests and provide faster and better service to the customers.

This architecture can also be scaled up easily and orchestration tools such as Kubernetes and Docker Swarms can help quickly deploy more servers in the cluster. These servers can be distributed globally based on traffic concentration and each cluster can employ load balancers and separate pipelining services.

We conclude this paper by demonstrating how our low cost architecture can help small and medium businesses based on IoT handle large number of requests at the same time, and provide faster responses to these API calls. We also discussed the scalability of this architecture and how server clusters can be formed based on traffic concentrations to keep performance optimal.

5.1. Future Scope and Enhancements

Our architecture can be molded an improved as newer technologies brings advancement in data processing, scalability, fault tolerance and in other aspects related to cloud. We mentioned the use of container orchestration tools such as Docker Swarm and Kubernetes in a server cluster. They enable “Desired State Management”, which takes a specific configuration and runs it across the entire infrastructure. Such tools are vital for scalability and improve deployment speeds.

One more current promising contender to enhance the platform is by using Blockchain technology. It is a revolutionary technology, which holds information in a shared and continually reconciled database, which is not stored in any one single location. The records they keep are truly public and easily verifiable, hosted by millions of computers simultaneously. This increases security to a completely new level, as transactions cannot be corrupted or altered, it would require overpowering the network.

REFERENCES

1. *Soo-Yeon Lee, Sa-rang Wi, Eunil Seo, Jun-Kwon Jung, Tai-Myuong Chung, “ProFiOt: Abnormal Behavior Profiling (ABP) of IoT Bevices based on Machine Learning Approach”, 2017 27th International Telecommunication Networks and Applications Conference (ITNAC)*
2. *Hafiz M. Farooq, Naif M. Otaibi, “Optimal Machine Learning Algorithms for Cyber Threat Detection”, 2018 UKSim-AMSS 20th International Conference on Computer Modelling and Simulation (UKSim)*
3. *Nathalie Baracaldo, Bryant Chen, Heiko Ludwig, Amir Safavi, Rui Zhang, “Detecting Poisoning Attacks on Machine Learning in IoT Environments”, 2018 IEEE International Congress on Internet of Things (ICIOT)*
4. *Sultan, J. A., & Jasim, R. M. (2016). Demand forecasting using artificial neural networks optimized by artificial bee colony. Int J. Manage, Inf. Technol. Eng, 4(7), 77-88.*
5. *Myungjun Son, Kyungyong Lee, “Distributed Matrix Multiplication Performance Estimator for Machine Learning Jobs in Cloud Computing”, 2018 IEEE 11th International Conference on Cloud Computing*

6. *Mazedur Rahman, Samira Iqbal, Jerry Gao, "Load Balancer as a Service in Cloud Computing", 2014 IEEE 8th International Symposium on Service Oriented System Engineering*
7. *BIRĂU, F. R. (2014). Forecasting financial time series based on Artificial Neural Networks. IMPACT: International Journal of Research in Business Management (IMPACT: IJRBM), ISSN (E).*
8. *Hero Handoko, Sani Muhamad Isa, S. Si, M. Kom, "High Availability Analysis with Database Cluster, Load Balancer and Virtual Router Redundancy Protocol", 2018 3rd International Conference on Computer and Communication Systems (ICCCS)*
9. *Xiwei Feng, Chuanying Jia, Jiaxuan Yang, "Semantic web-based publish-subscribe system", 2008 IEEE International Conference on Service Operations and Logistics, and Informatics*
10. *R.G.J. Wijnhoven, P.H.N. de With, "Fast Training of Object Detection using Stochastic Gradient Descent", 2010 20th International Conference on Pattern Recognition*
11. *Guojing Cong, Onkar Bharadwaj, "A Hierarchical, Bulk-Synchronous Stochastic Gradient Descent Algorithm for Deep-Learning Applications on GPU Clusters", 2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA)*
12. *Hung-Leng Chen, Kun-Ta Chuang, Ming-Syan Chen, "On Data Labelling for Clustering Categorical Data", 2008 IEEE Transactions on Knowledge and Data Engineering (Volume 20, Issue 11)*
13. *Venkateshwara Reddy Eluri, M. Ramesh, Amina Salim Mohd Al-Jabri, "A comparative study of various clustering techniques on big data sets using Apache Mahout", 2016 3rd MEC International Conference on Big Data and Smart City (ICBDSC)*
14. *Xue Zhang, Dong-yan Zhao, Li-wen Chen, Wang-hua Min, "Batch Mode Active Learning Based Multi-view Text Classification", 2009 6th International Conference on Fuzzy Systems and Knowledge Discovery*
15. *Jing Wang, Yue Wang, Wei Wang, Huatong Wie, Liulin Cao, Qibing Jin, "Adaptive terminal iterative learning for batch process with batch-varying parameters", Proceeding of the 32nd Chinese Control Conference*